

「CCFinderNexGenの開発」

神谷年洋
産業技術総合研究所
2004年度IPA米路ソフトウェア想像事業開発者

2005/08/18

IPA米路事業 梅村PM研究成果報告会

1

コードクローンとは

- コードクローンとは、ソースコード中の全く同一のコードの断片、または、類似したコード断片のこと
 - 開発者がソースコードをコピー&ペーストするなどの原因により、ソースコード中に作り込まれます
- コードクローンはソフトウェアの保守を困難にする
 - コードクローンが存在すると、あるコード断片にバグが見つかったとき、そのコードクローンが100個あれば、その100個のコード断片をすべて探し出し修正する必要があります。
 - 特に、大規模なソースコードにおいては、手作業でコード断片を見つけ出すことは困難
- コードクローンへの対策は、除去、予防、管理

2005/08/18

IPA米路事業 梅村PM研究成果報告会

2

コードクローン検出

- コードクローンへの対策には、まずコードクローンの存在を知ることが必要
- 大規模なソースコードから効果的にコードクローンを検出するための手法はほとんど無かった
 - たとえば、grepなどのツールは、コピー&ペーストされた後、変数名や関数名が修正されたコードを検索するためには、手作業でパターンを記述することが必要
 - ASTなどの抽象度が高いデータ構造を利用した解析は、スケーラビリティが高くないために、大規模なソースファイルの分析には向かない

2005/08/18

IPA米路事業 梅村PM研究成果報告会

3

CCFinderとGemini

- CCFinder(2000)とは
 - コードクローン検出ツール
 - ソースコード全体を一度に読み込み、互いに同一または類似したコード断片(すなわち、コードクローン)となっている場所をすべて探し出し、その場所を出力する
 - バグが発見された後でそのコードクローンを探す
 - あらかじめコードクローンとなっている部分をすべて探し出してコメントに書き込んでおく
 - 当初から大規模なソースコードに適用することを想定し、数百万行規模のソースコードに対する安定した運用実績を持っている
 - 対象となるプログラミング言語に応じたプリプロセスを行い
 - 変数名や関数名の違いを吸収する
 - {}などの表記の揺れを吸収する

2005/08/18

IPA米路事業 梅村PM研究成果報告会

4

□ Gemini(2002)とは

- コードクローン検出結果を散布図やメトリックグラフなどを用いて対話的に分析するためのGUIツール
 - ソースコードのどの部分にコードクローンが多く作り込まれているか、
 - 除去することで高い改善の効果が得られそうなコードクローンはどれかといった分析を行う

2005/08/18

IPA米路事業 梅村PM研究成果報告会

5

□ CCFinder/Geminiの利用

- CCFinder/Geminiはこれまでに産業界、学界に配布されてきており、
 - 産業界においては、リファクタリングのため、あるいは、ソースコードの品質の調査のために使われている
 - 学界においては、CCFinder自身が研究として発表されており、さらにCCFinderによって実験を行った研究がいくつも発表されている
- コードクローン検出ツールの一標準
 - 新しいコードクローン検出ツールが発表されるときにCCFinderとの性能比較が行われるなど

2005/08/18

IPA米路事業 梅村PM研究成果報告会

6

「CCFinderNexGenの開発」

CCFinderXとは

- 従来のCCFinderにおいて問題となった点(カスタマイズ性、コードクローンの絞込み)を解決すべく、新設計に基づいてスクラッチから開発されたコードクローン検出ツール

特徴

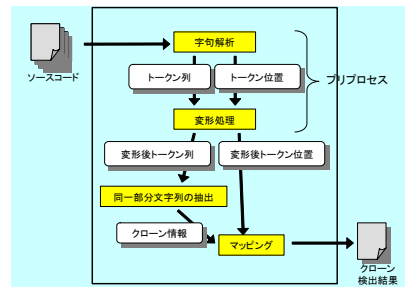
- プリプロセス処理の利用者によるカスタマイズ
 - プログラミング言語の方言への対応や、新たなプログラミング言語への対応を可能にした
- プログラムパターンによるフィルタリング
- スケーラビリティ、処理速度の改善(繰り返し分析を行う際の)

2005/08/18

IPA未踏事業 梅村PM研究成果報告会

7

処理の流れ(旧CCFinder)

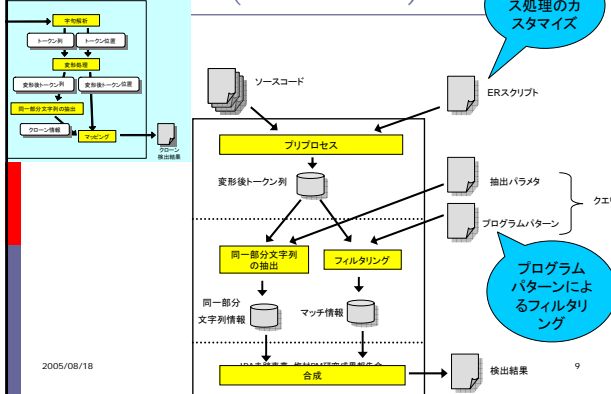


2005/08/18

IPA未踏事業 梅村PM研究成果報告会

8

処理の流れ(CCFinderX)



2005/08/18

9

技術開発

新しいコードクローン検出アルゴリズム

- 接辞尾木を用いないことでピークメモリ使用量を削減

プリプロセス用の専用パーサー

- 頑健さ
 - エラーを含むソースファイルを扱う
- 記述の容易さ
 - 正規表現のような記述から書き始められて、いつの間にかツリーのような構造を築ける
 - 途中の処理が見える(デバッグにおいて)
 - はじめから完全なツリーを作るのではなく、ツリーの構造を作り変える操作が中心となる

GUIフロントエンド

- スケーラビリティを確保
 - 中間データをファイル上を持つ
- SWTを利用して、Eclipseへの組み込みを目指す
- 機能が増えたことに伴い、「スコープ」を導入して操作体系を一新・整理した

2005/08/18

IPA未踏事業 梅村PM研究成果報告会

10

プリプロセス用の専用パーサー

- 現実のソースコードを相手にするためには、
 - プログラミング言語の方言を用いている
 - エラーが含まれている
 - 処理系によって異なるコードに化ける(C言語のマクロなど)
- →不完全なソースコードを前提としたプリプロセスが必要

開発目標

- 頑健さ
 - エラーを含むソースファイルを扱う
- 記述の容易さ
 - 正規表現のような記述から書き始められて、いつの間にかツリーのような構造を築ける
 - 途中の処理が見える(デバッグにおいて)
 - はじめから完全なツリーを作るのではなく、ツリーの構造を作り変える操作が中心となる

2005/08/18

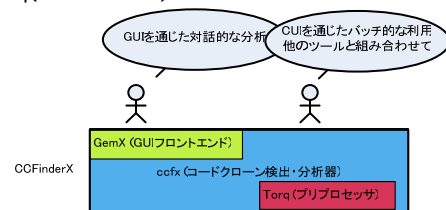
IPA未踏事業 梅村PM研究成果報告会

11

CCFinderXの構成

CCFinderXパッケージを構成するコンポーネント

- GemX(GUIフロントエンド)、
- Ccfx(コードクローン検出・分析器)、
- Torq(プリプロセッサ)



2005/08/18

IPA未踏事業 梅村PM研究成果報告会

図4.CCFinderXの構成

12

TODO(積み残し)

- ccfx
 - 移植
 - とりあえず Darwin
 - 日本語文字を扱えるようにする
 - ファイル名、ソースコード中の文字
 - プリプロセススクリプトの拡充
 - Java, COBOL用
 - 下位互換サポート
 - CCFinder7のファイルを生成できるように
 - ギャップ・クローン検出
- GemX
 - ccfxの機能をすべて使えるようにする
 - ソースファイルを2つ並べて比較する
 - その他

2005/08/18

IPA未踏事業 梅村PM研究成果報告会

13

まとめ

- 新しいコードクローン検出ツールCCFinderXを開発した
 - ほぼ当初の予定通りの機能を実装した
 - スケーラビリティ、処理速度の改善
 - スクリプトによるプリプロセス
 - プログラムパターンによるフィルタリング
- 新しいGUIフロントエンドを開発した
 - 開発が必要になった理由
 - CCFinderXでは機能が大幅に増えたため
 - GUIフロントエンドのスケラビリティを確保するため
 - Eclipseへの組み込みを目指すため

2005/08/18

IPA未踏事業 梅村PM研究成果報告会

14

新旧CCFinder比較

機能	CCFinder7	CCFinderX	備考
フォーマット	クローンペア クローンセット	format7- format7-	formatX formatX
基本オプション	ccreformer オプション-b オプション-i オプション-o オプション-m	○ ○ ○ ○	不要 クローンの最小長 入力ファイル指定 メモリヒント
プリプロセス	オプション-r *tora ccfx_prep_scripts.ini オプション-k	○ x ○ ○	プリプロセスオプション ユーザー定義プリプロセス 括弧子からのプログラミング言語判定 繰り返し部分検出
ファイルグループオプション	-c "-ng"	○ ○	x x ファイル内、ファイル間、グループ間
対応プログラミング言語	plaintext, C/C++, Java, COBOL, FORTRAN	○	C/C++ CCFinderXの対応言語は順次追加予定
パフォーマンス	linux-2.6.11のソースへの適用実験	40分	30分(1回目) / 7分(2回目) CPU Celeron 1.4GHz, メモリ1.21GB

2005/08/18

IPA未踏事業 梅村PM研究成果報告会

15